Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR in the Wild

TRACE-CS: A Hybrid Logic–LLM System for Explainable Course Scheduling

Stylianos Loukas Vasileiou[1,2] and **William Yeoh**[2]
[1]New Mexico State University
[2]Washington University in St. Louis
stelios@nmsu.edu, wyeoh@wustl.edu

Abstract

We present TRACE-CS, a novel hybrid system that combines symbolic reasoning with large language models (LLMs) to address contrastive queries in course scheduling problems. TRACE-CS leverages logic-based techniques to encode scheduling constraints and generate provably correct explanations, while utilizing an LLM to process natural language queries and refine logical explanations into user-friendly responses. This system showcases how combining symbolic KR methods with LLMs creates explainable AI agents that balance logical correctness with natural language accessibility, addressing a fundamental challenge in deployed scheduling systems.

1 Introduction

Scheduling systems, which allocate finite resources to multiple agents over time, are ubiquitous in real-world environments, from personnel shift assignments (Van den Bergh et al. 2013) to Mars rover activities (Chi, Chien, and Agrawal 2020). Beyond generating valid and optimal schedules, it is crucial to ensure that both the schedule and the decision-making process are *explainable* to human users. *Explainable scheduling*, therefore, is essential for understanding scheduling decisions, rectifying issues, and providing explanations for specific decisions or schedule generation failures. Most of the work in this space have relied on symbolic, logical methods that generate valid and sound explanations.

At the other end of the spectrum, the emergence of large language models (LLMs) has marked a significant milestone in AI. While LLMs excel at generating coherent and contextually relevant text (Brown et al. 2020), their reliance on statistical inference leads to challenges in maintaining logical consistency and accuracy in reasoning and planning tasks (McCoy et al. 2023; Valmeekam et al. 2023). This limitation is particularly apparent when explanations need to be both linguistically coherent and logically sound. In contrast, symbolic, logical methods provide a robust medium for reasoning and planning due to their ability to perform valid and sound inference. This realization offers an opportunity to combine the strengths of both LLMs and symbolic methods, creating synergistic systems that ensure decisions are not only provably correct and robust, but also communicated in a user-friendly manner.

In this paper, we present **TRACE-CS** (*Trustworthy ReAsoning for Contrastive Explanations in Course Scheduling Problems*), a synergistic system that combines symbolic reasoning with the natural language capabilities of LLMs for generating explanations in course scheduling problems. Particularly, TRACE-CS generates natural language explanations for contrastive user queries (e.g., "Why course X instead of course Y?") by leveraging a state-of-the-art symbolic explainer (Vasileiou, Previti, and Yeoh 2021) together with an LLM-powered user interface for natural language interactions, thus ensuring that the explanations are provably trustworthy as well as communicated to users in a natural format.

In short, this paper focuses on the practical implementation, deployment, and evaluation of TRACE-CS as a case study in hybrid KR systems. We demonstrate how symbolic methods provide correctness guarantees while LLMs enhance user experience through natural language processing, creating a synergistic system with real-world utility. Our experimental results quantify these benefits, showing perfect accuracy in explanations while maintaining natural language accessibility—a significant improvement over LLM-only approaches.

2 Related Work

Explainable scheduling research has predominantly relied on logical symbolic methods (Cyras et al. 2019; Agrawal, Yelamanchili, and Chien 2020; Bertolucci et al. 2021; Pozanco et al. 2022; Powell and Riccardi 2022; Vasileiou et al. 2022; Vasileiou, Xu, and Yeoh 2023; Zehtabi et al. 2024). While grounded in sound inference procedures, these approaches often produce explanations that are difficult to communicate to users due to their logic-based nature. Attempts to mitigate this limitation have used templates mapping logical explanations to pre-specified natural language sentences (Pozanco et al. 2022; Vasileiou, Xu, and Yeoh 2023) or visualization interfaces (Čyras, Lee, and Letsios 2021; Kumar et al. 2022; Powell and Riccardi 2022).

Concurrently, LLMs have revolutionized natural language processing and found applications across diverse domains, including planning (Kambhampati et al. 2024), code generation (Roziere et al. 2023), and medical applications (Zhou et al. 2023). However, the integration of LLMs with symbolic explainable scheduling systems remains largely unex-

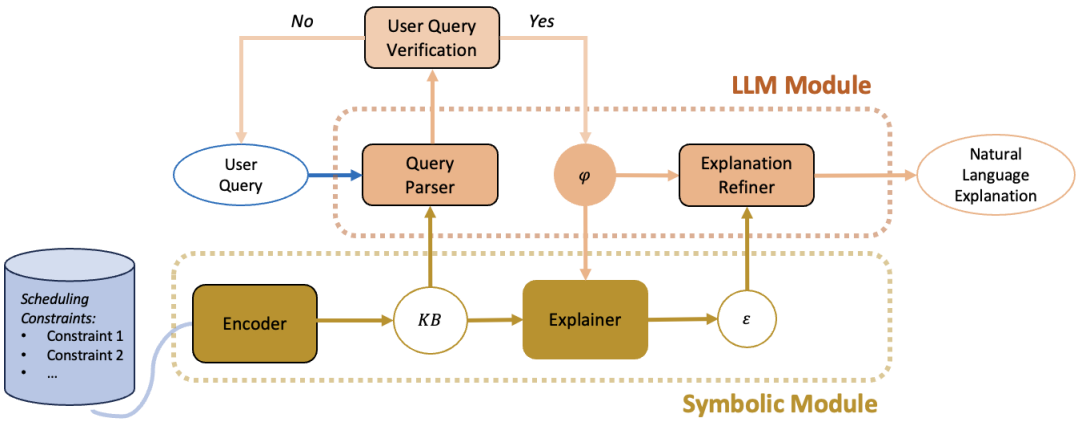

Figure 1: The TRACE-CS workflow.

plored. Our work, TRACE-CS, represents the first attempt to address this gap by presenting a novel hybrid system that synergistically combines a symbolic explainable scheduling module with an LLM module.

3 TRACE-CS System Architecture Overview

The TRACE-CS system architecture, illustrated in Figure 1, consists of two primary components: a *Symbolic Module* handling constraint encoding and explanation generation, and an *LLM Module* managing natural language interaction. The workflow is as follows: (1) The user submits a contrastive query in natural language; (2) The Query Parser extracts the information from the query and converts it into a logical representation φ consistent with the knowledge base KB created by the Encoder; (3) The user verifies if the extracted query information corresponds to the original query, and proceeds to the next step if it is; (4) The Explainer generates a symbolic explanation ϵ for φ with respect to KB; (5) The Explanation Refiner converts ϵ into natural language and outputs it to the user.

3.1 Symbolic Module

The Symbolic Module forms the core of TRACE-CS and consists of two subcomponents, the *Encoder*, which encodes the scheduling constraints into a logical knowledge base, and the *Explainer*, which generates minimal explanations for user queries with respect to the knowledge base.

Encoder. The Encoder transforms course scheduling constraints into a Boolean satisfiability (SAT) problem.[1] The system models scheduling decisions using Boolean variables and logical clauses derived from degree requirements and university policies. Specifically, for each course c and semester s, the system creates a course variable $\text{var}(c, s)$ that indicates whether course c is scheduled in semester s. The constraints span several categories, such as degree requirements (e.g., core courses, elective distributions, total credit requirements), temporal constraints (e,g., prerequisites, semester credit limits), and general scheduling constraints (e.g., each selected course is assigned to exactly one semester). For example, the prerequisite constraint 'YNP H57 must be completed before XOX R89" is encoded as the following logical clause: $\neg\text{var}(\text{XOX\_R89}, s) \vee \bigvee_{t=0}^{s-1} \text{var}(\text{YNP\_H57}, t)$, where s is a semester that XOX R89 could be scheduled, and $t < s$. This ensures that if XOX R89 is scheduled in semester s, then YNP H57 must be scheduled in some previous semester $t < s$.

Explainer. The Explainer generates a minimal explanation for the user contrastive query φ (processed by the LLM module) using the logic-based explanation generation algorithm from (Vasileiou, Previti, and Yeoh 2021; Vasileiou, Xu, and Yeoh 2023). In essence, the algorithm takes as input the KB and the query φ, where $\text{KB} \models \varphi$, and outputs a set of logical clauses $\epsilon \subseteq \text{KB}$ such that $\epsilon \models \varphi$, and $\nexists \epsilon' \subset \epsilon$ such that $\epsilon' \models \varphi$.[2] In other words, it outputs a $\subseteq$-minimal explanation ϵ.

3.2 LLM Module

The LLM Module serves as the interface between the user and the Symbolic Module, handling natural language processing tasks through two subcomponents: the *Query Parser*, which interprets contrastive queries and converts them into logical representations, and the *Explanation Refiner*, which translates logical explanations into user-friendly natural language responses.

Query Parser. The Query Parser converts natural language contrastive queries into logical representations φ compatible with the knowledge base KB. The parser uses an LLM

[1] A plethora of scheduling problems has been modeled using SAT-based approaches (Crawford and Baker 1994; Pinto and Grossmann 1997; Kundu and Acharyya 2008; Ansótegui et al. 2011; Haspeslagh et al. 2013; Bofill et al. 2015; Demirović, Musliu, and Winter 2019). In such problems, a schedule is found if and only if the encoded KB has a satisfying model.

[2] Practically, the algorithm leverages the fact that if $\text{KB} \models \varphi$, then $\text{KB} \wedge \neg\varphi \models \bot$, and uses SAT-based solvers optimized to find minimal unsatisfiable sets (MUSes) and minimal correction sets (MCSes) (Marques-Silva 2012; Marques-Silva et al. 2013).

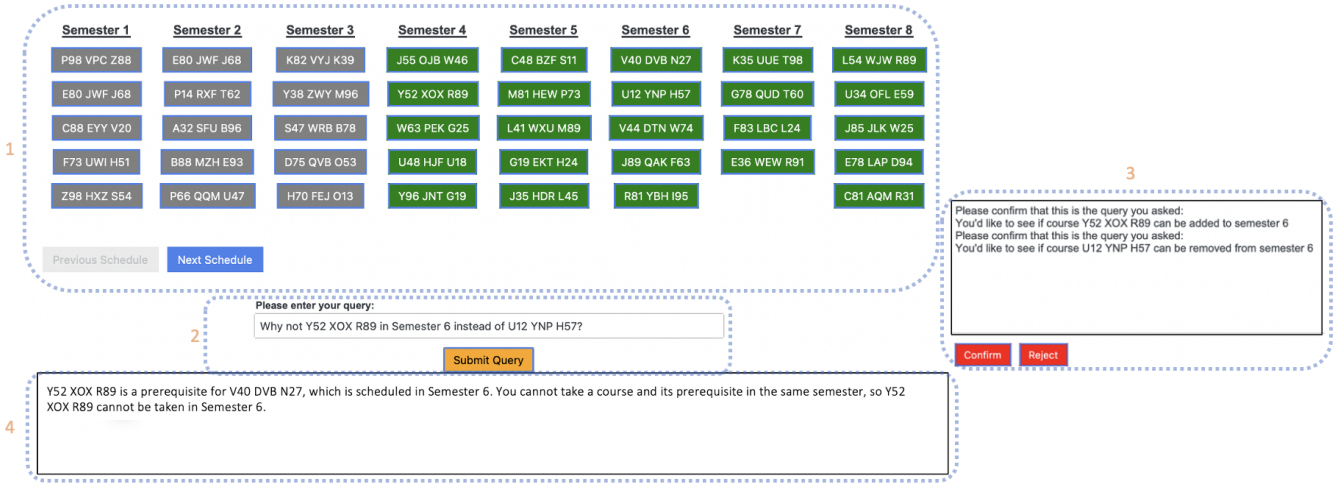

Figure 2: The TRACE-CS interface showing the explanation workflow: (1) course schedule display across 8 semesters; (2) user query input; (3) query verification step with user confirmation; and (4) generated explanation output.

with in-context learning to extract three key components from user queries: (1) course names, (2) target semesters, and (3) conditions (positive or negative). For example, the query "Why not XOX R89 instead of YNP H57?" is parsed to extract two courses: XOX R89 with a positive condition (should be scheduled) and YNP H57 with a negative condition (should not be scheduled), both targeting the semester where YNP H57 is currently scheduled.[3] The extracted information is then converted into clauses that can be evaluated against the knowledge base. Finally, as recent work highlights potential limitations of LLMs in formal interpretation tasks (Karia et al. 2024), TRACE-CS includes a user verification step before proceeding to explanation generation to ensure that the converted queries are correct.

Explanation Refiner. The Explanation Refiner takes the symbolic explanation ϵ from the Explainer and translates it into coherent natural language responses. For each clause in ϵ, the system retrieves the corresponding English label—a short sentence describing what the constraint means in natural language. The refiner then uses an LLM with in-context learning to process these labels along with contextual information about the current schedule and course descriptions, generating a coherent explanation while maintaining logical accuracy. For instance, multiple related constraints about prerequisite violations are grouped and presented as a single coherent explanation rather than as separate constraint statements.

4 Proof-of-Concept

We implemented TRACE-CS as a proof-of-concept for undergraduate computer science course scheduling at our university.[4] The system handles scheduling decisions across eight academic semesters, incorporating real course data and degree requirements from the university's official sources.

Domain and Data Collection. The application domain involves scheduling courses for a Bachelor of Science in Computer Science degree, which requires 120 credit hours distributed across core courses, electives, and general education requirements. We collected course data by scraping our university's official course catalog and degree requirements, extracting course codes, credit hours, prerequisites, and course descriptions. The dataset includes over 200 courses spanning core CS courses, CS electives, science electives, and social science/humanities requirements. All courses are anonymized for the blind review process. Prerequisites form complex dependency chains–for instance, WJW R89 (Analysis of Algorithms) requires XOX R89 (Data Structures), which in turn requires VPC Z88 (Introduction to Computer Science).

Implementation. The system is implemented in Python, with the Symbolic Module using the PySAT library (Ignatiev, Morgado, and Marques-Silva 2018) for SAT encoding and solving, and the LLM Module using GPT-4.1 (OpenAI 2023) for natural language processing. The encoder generates constraints for degree requirements (e.g., "at least 45 CS elective credits"), temporal dependencies (e.g., prerequisite chains), and scheduling logistics (e.g., 9-15 credits per semester). The system produces multiple valid schedules using solution blocking techniques, allowing users to explore different scheduling options.

User Interface. Figure 2 shows the interface, which displays the generated schedule as a semester-by-semester course layout. Users can submit contrastive queries through a text input field, such as "Why not XOX R89 in semester 6 instead of YNP H57?". The interface includes a verification step where users confirm that the system correctly parsed their query before proceeding to explanation generation. This verification step addresses potential limitations

[3]The parser includes examples in the prompt to handle various query formats and performs fuzzy matching for partially specified course names.

[4]Code repository: https://github.com/YODA-Lab/TRACE-CS.

Complexity Level	Accuracy (%)		Avg. Words		Avg. Runtime (sec)	
	TRACE-CS	GPT-4.1	TRACE-CS	GPT-4.1	TRACE-CS	GPT-4.1
1	100.0	62.0	64.7	129.8	11.0	3.9
2	100.0	56.0	63.0	152.1	12.2	4.5
4	100.0	52.0	102.7	165.3	15.9	4.2
6	100.0	46.5	122.4	190.5	20.0	4.7
Overall	**100.0**	**54.1**	**81.8**	**159.4**	**14.7**	**4.3**

Table 1: Comparative evaluation results between TRACE-CS and GPT-4.1 approach across 550 queries of various complexity levels.

in LLM query interpretation and ensures user intent is accurately captured.

Query Types and System Response. The system handles various contrastive query patterns, including single-course questions (e.g., "Why XOX R89?"), temporal queries ("Why not XOX R89 in semester 5?"), and comparative queries ("Why LAP D94 instead of UUE T98 in Semester 6?"). For each query, the system identifies a minimal set of constraints preventing the alternative and presents explanations such as "WJW R89 cannot be scheduled because its prerequisite XOX R89 has not been completed" or "The total credits for CS electives must sum to 45 credits." The interface maintains conversation history, allowing users to ask follow-up questions about the same schedule.

4.1 Computational Evaluation

We conducted a comparative evaluation of TRACE-CS against a pure LLM-only approach using GPT-4.1.[5] The evaluation used 550 contrastive queries across several different course schedules, and was run on a machine with an M1 Max processor and 32GB of RAM.

Experimental Setup. We generated queries of varying complexity levels, where complexity indicates the number of courses mentioned in the query (e.g., "Why not YNP H57?" has complexity 1, while "Why VPC Z88 in semester 1 and JWF J68 in semester 2?" has complexity 2). For the LLM-only baseline, we provided GPT-4.1 with the course schedule, course descriptions, and all scheduling constraints, asking it to generate explanations directly without the logical reasoning component. We also provided it with a few example queries and their correct corresponding explanations.

Evaluation Metrics. We measured three key aspects: (1) *explanation correctness* with respect to the scheduling constraints, evaluated manually by the authors, (2) *verbosity* measured by word count in generated explanations, and (3) *response time* for explanation generation.

Results. Table 1 shows the results. TRACE-CS achieved 100% correctness across all complexity levels, while GPT-4.1 achieved only 54.1% correctness overall, with performance ranging from 62.0% for queries of complexity 1 to 46.5% for queries of complexity 6. The low accuracy indicates a limitation of pure LLM approaches for logical reasoning tasks. In terms of verbosity, GPT-4.1 generated significantly longer explanations, averaging 159.4 words compared to TRACE-CS's 81.8 words. This verbosity increased substantially with query complexity, reaching 190.5 words for complexity 6 queries compared to TRACE-CS's 122.4 words. It is worth noting that GPT-4.1 exhibited a tendency to generate non-minimal explanations that included most or all applicable constraints rather than identifying the specific minimal set causing the conflict, despite being prompted to only generate the most relevant and minimal reasons. While these comprehensive explanations may be technically correct in some cases, they might overwhelm users as they include unnecessary information.

For response time, as expected, GPT-4.1 was faster, averaging 4.3 seconds compared to TRACE-CS's 14.7 seconds. However, TRACE-CS's additional computational cost (due to calling SAT solvers) leads to perfect accuracy of the explanation. Overall, the results reveal a critical trade-off between speed and reliability in explanation generation. GPT-4.1's poor performance possibly stems from its statistical inference approach, which struggles with the precise logical reasoning required for constraint satisfaction problems. In contrast, TRACE-CS leverages symbolic reasoning to guarantee logical correctness while using the LLM component solely for natural language processing tasks where it excels.

5 Conclusions

We presented TRACE-CS, a hybrid system that combines logical reasoning with LLMs for explainable course scheduling. Our evaluation demonstrates that TRACE-CS achieves perfect logical correctness while generating concise, minimal explanations–substantially outperforming the pure LLM approach that achieved only 54.1% accuracy. As LLM capabilities continue to evolve, the modular design of TRACE-CS might provide a framework for incorporating improved reasoning models while maintaining the guarantee of logical correctness through symbolic verification.

It is important to note that our evaluation focused on a single LLM (i.e., GPT-4.1). Recent advances in reasoning capabilities of LLMs suggest that newer or more specialized models might achieve better performance on logical reasoning tasks. Models specifically trained on formal reasoning or those with enhanced chain-of-thought capabilities could potentially narrow the gap with symbolic approaches. Evaluating the system with newer models and/or domain-specific reasoning models would provide insights into the evolving landscape of neural reasoning capabilities. Additionally, our evaluation used a straightforward prompting strategy; more sophisticated prompting techniques, such as structured reasoning prompts or multi-step verification processes, might improve LLM performance as well.

Finally, while demonstrated on course scheduling, the hybrid architecture of TRACE-CS can extend naturally to other constraint-based domains, such as planning and resource allocation. Moreover, conducting user studies to assess explanation quality from an end-user perspective would provide valuable insights into the practical utility of minimal versus comprehensive explanations in real-world deployment scenarios.

[5] We chose GPT-4.1 because it was one of the best performing and most affordable model at the time of writing this paper.

Acknowledgements

Stylianos Loukas Vasileiou and William Yeoh are partially supported by the National Science Foundation (NSF) under award 2232055. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or governments.

References

Agrawal, J.; Yelamanchili, A.; and Chien, S. 2020. Using explainable scheduling for the Mars 2020 rover mission. *arXiv preprint arXiv:2011.08733*.

Ansótegui, C.; Bofill, M.; Palahí, M.; Suy, J.; and Villaret, M. 2011. Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem. In *Proceedings of the Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2–9.

Bertolucci, R.; Dodaro, C.; Galatà, G.; Maratea, M.; Porro, I.; and Ricca, F. 2021. Explaining ASP-based operating room schedules. In *Proceedings of the Workshop on Explainable Logic-Based Knowledge Representation*.

Bofill, M.; Garcia, M.; Suy, J.; and Villaret, M. 2015. MaxSAT-based scheduling of B2B meetings. In *Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, 65–73.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language models are few-shot learners. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 1877–1901.

Chi, W.; Chien, S.; and Agrawal, J. 2020. Scheduling with complex consumptive resources for a planetary rover. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 348–356.

Crawford, J., and Baker, A. 1994. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1092–1097.

Cyras, K.; Letsios, D.; Misener, R.; and Toni, F. 2019. Argumentation for explainable scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2752–2759.

Čyras, K.; Lee, M.; and Letsios, D. 2021. Schedule explainer: An argumentation-supported tool for interactive explanations in makespan scheduling. In *Proceedings of the International Workshop on Explainable and Transparent AI and Multi-Agent Systems*, 243–259.

Demirović, E.; Musliu, N.; and Winter, F. 2019. Modeling and solving staff scheduling with partial weighted MaxSAT. *Annals of Operations Research* 275:79–99.

Haspeslagh, S.; Messelis, T.; Berghe, G. V.; and De Causmaecker, P. 2013. An efficient translation scheme for representing nurse rostering problems as satisfiability problems. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, 303–310.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python toolkit for prototyping with SAT oracles. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 428–437.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and B Murthy, A. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 22895–22907.

Karia, R.; Dobhal, D.; Bramblett, D.; Verma, P.; and Srivastava, S. 2024. ∀uto∃val: Autonomous assessment of llms in formal synthesis and interpretation tasks. *arXiv preprint arXiv:2403.18327*.

Kumar, A.; Vasileiou, S. L.; Bancilhon, M.; Ottley, A.; and Yeoh, W. 2022. VizXP: A visualization framework for conveying explanations to users in model reconciliation problems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 701–709.

Kundu, S., and Acharyya, S. 2008. Stochastic local search approaches in solving the nurse scheduling problem. In *Proceedings of the Computer Information Systems and Industrial Management Applications (CISIM)*, 202–211.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On computing minimal correction subsets. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 615–622.

Marques-Silva, J. 2012. Computing minimally unsatisfiable subformulas: State of the art and future directions. *Journal of Multiple-Valued Logic & Soft Computing* 19.

McCoy, R. T.; Yao, S.; Friedman, D.; Hardy, M.; and Griffiths, T. L. 2023. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Pinto, J., and Grossmann, I. 1997. A logic-based approach to scheduling problems with resource constraints. *Computers & Chemical Engineering* 21(8):801–818.

Powell, C., and Riccardi, A. 2022. Abstract argumentation for explainable satellite scheduling. In *Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10.

Pozanco, A.; Mosca, F.; Zehtabi, P.; Magazzeni, D.; and Kraus, S. 2022. Explaining preference-driven schedules: The EXPRES framework. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 710–718.

Roziere, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Remez, T.; Rapin, J.; et al.

2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 75993–76005.

Van den Bergh, J.; Beliën, J.; De Bruecker, P.; Demeulemeester, E.; and De Boeck, L. 2013. Personnel scheduling: A literature review. *European Journal of Operational Research* 226(3):367–385.

Vasileiou, S. L.; Yeoh, W.; Son, T. C.; Kumar, A.; Cashmore, M.; and Magazzeni, D. 2022. A logic-based explanation generation framework for classical and hybrid planning problems. *Journal of Artificial Intelligence Research* 73:1473–1534.

Vasileiou, S. L.; Previti, A.; and Yeoh, W. 2021. On exploiting hitting sets for model reconciliation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 6514–6521.

Vasileiou, S. L.; Xu, B.; and Yeoh, W. 2023. A logic-based framework for explainable agent scheduling problems. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2402–2410.

Zehtabi, P.; Pozanco, A.; Bolch, A.; Borrajo, D.; and Kraus, S. 2024. Contrastive explanations of centralized multi-agent optimization solutions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 671–679.

Zhou, H.; Gu, B.; Zou, X.; Li, Y.; Chen, S. S.; Zhou, P.; Liu, J.; Hua, Y.; Mao, C.; Wu, X.; et al. 2023. A survey of large language models in medicine: Progress, application, and challenge. *arXiv preprint arXiv:2311.05112*.